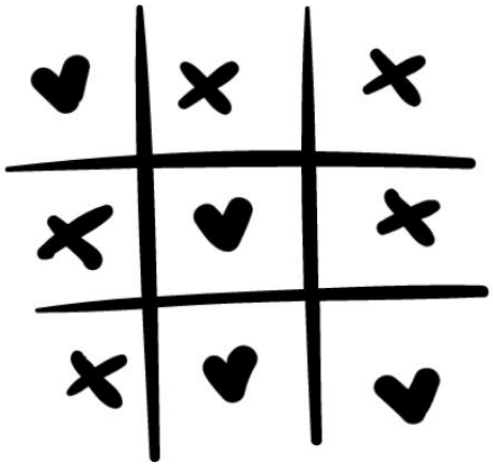
TIC-TAC-TOE



Presented to:
Dr. Howida Abd AlLatif

Presented by:

مريم عادل عبدالعظيم عبدالله

مريم محمد السيد عبدالعزيز

نادين أحمد محمد الهادي

نادين هيثم فتح الله عبدالحفيظ

هنا أحمد محمد الهادي

Contents:

Introduction:

Tic-Tac-Toe is a classic two-player strategy game played on a 3x3 grid. Each player alternates turns, marking a cell with their respective symbol (X or O), aiming to align three of their symbols consecutively in a row, column, or diagonal. The game's simple mechanics make it an ideal candidate for demonstrating logical problem-solving and rule-based systems.

In this context, Tic-Tac-Toe is treated as a pure logic game, where every aspect of its functionality is implemented through logical design principles. The objective is to create a system that manages all aspects of the game, including tracking player turns, validating moves, detecting winning or draw conditions, and ensuring proper state transitions.

The problem focuses on designing a system that operates entirely using combinational and sequential logic, without relying on external algorithms or software. This approach highlights the fundamental principles of logic design, such as state diagrams, truth tables, and equations to model and control the game's behavior. The exercise not only demonstrates how logical operations can govern a complete system but also emphasizes the precision and accuracy required in pure logic implementations.

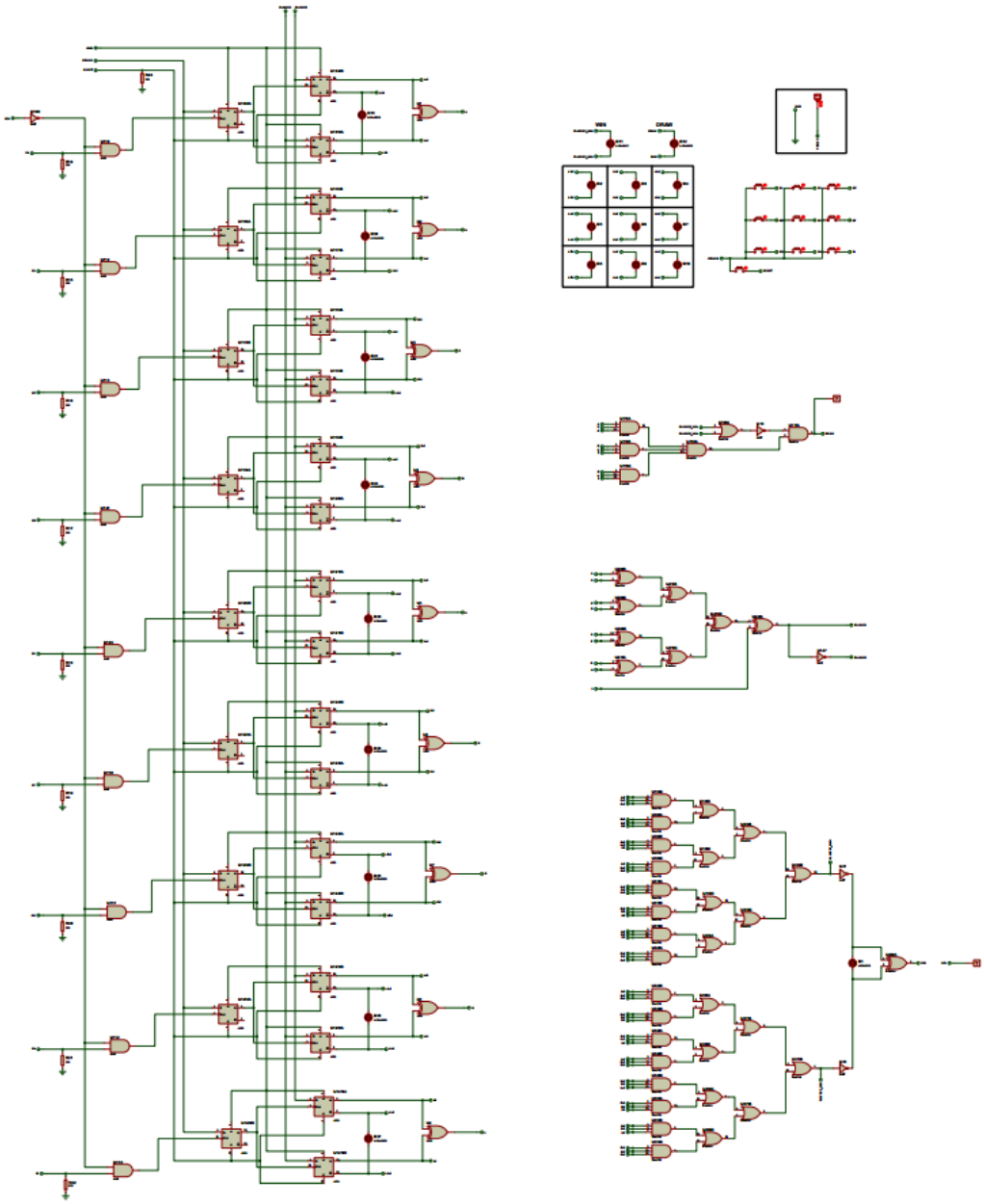
To ensure proper functionality, the following requirements must be considered:

- **Protect Player Moves:** Once a player selects a cell, it must remain locked and cannot be modified by the other player. For example, if Player 1 marks the first cell, Player 2 should not be able to select or change it.
- **Game Reset Option:** A reset button must be provided, allowing players to restart the game at any time. This should clear all moves and return the game to its initial state.
- **Result State Locking:** When a player wins, the result must be fixed and cannot be altered. For example, if Player 1 wins and Player 2 makes another move that would meet the winning criteria, the game must still display "Player 1" as the winner without updating the outcome.

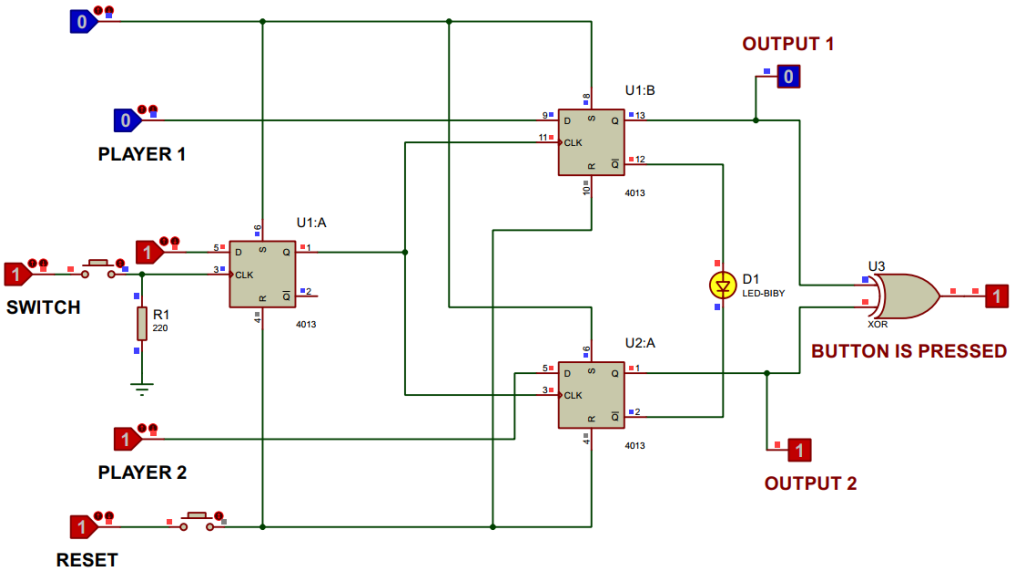
Circuit design:

The circuit implements the Tic Tac Toe game logic using combinational logic components and is divided into the following key modules:

1. Button Circuit: Handles the player inputs and ensures proper turn-taking during the game.
2. Profile Generation: Determines which player's turn it is and generates the corresponding control signals for further processing.
3. Winning Circuit: Identifies a winner by evaluating the game board state based on predefined winning conditions for rows, columns, and diagonals.
4. Checking Draw Circuit: Verifies if the game ends in a draw by ensuring no winning conditions are met and all positions are filled.



Button Circuit:

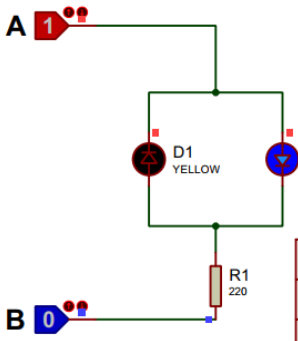


The first step in our design was configuring the switch for the LEDs. We needed to ensure that pressing a button would light up the corresponding LED. Additionally, the LEDs should alternate between colors, depending on which player's turn it is.

One challenge we encountered was that the button states would reset immediately after being pressed, causing the information to be lost. To address this, we implemented a D flip-flop circuit to store the button state.

In our design, the input flip-flop is connected to a constant high value. When a button is pressed, the signal transitions from 0 to 1, which triggers two additional flip-flops. These flip-flops capture and hold their respective values, and their direct outputs are fed into an XOR gate for further processing, while their inverts are fed into a bi-color LED to light it up.

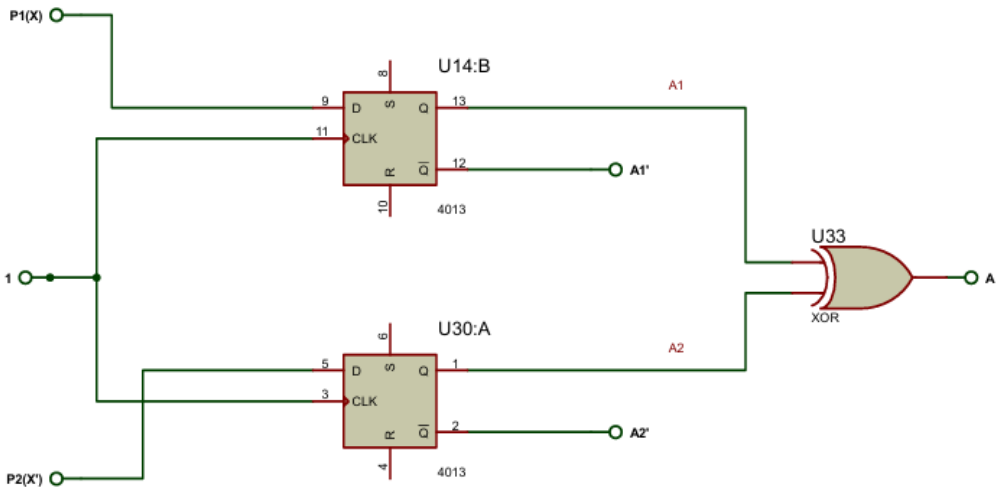
The led lights up blue if PLAYER-1 = 1, and PLAYER-2 = 0, yellow if PLAYER-1 = 0 ,and PLAYER-2 =1



	A	B
OFF	0	0
YELLOW	0	1
BLUE	1	0
OFF	1	1

The third flip-flop's main function is to make sure that LED won't change its color no matter how many times we press it, this happens because the 2 flip-flops on the right can have their positive trigger only once. This is possible because the left flip-flop has constant input 1 and won't go from 0 to 1 again.

- **Simplified Representation:**



- **State Equations:**

X = Player1

X' = Player2

Q+ = D

A1+ = X

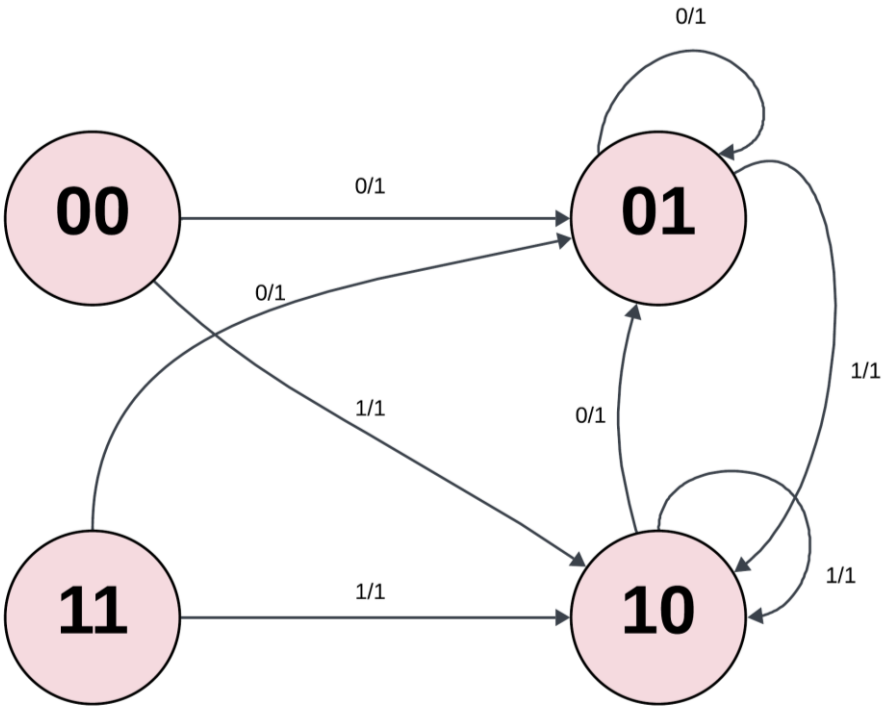
A2+ = X'

- **State Table:**

A1	A2	X	A1+	A2+	A
0	0	0	0	1	1
0	0	1	1	0	1
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	1	0	1

Notice that the output of the XOR gate (A) is always equal to 1. This means that the button has been pressed, regardless of whose turn it is.

- **State Diagram:**



The state diagram clearly shows that all states transition to either 10 or 01. These are the only possible states in the game, as it is impossible for both players to take their turn simultaneously or for neither player to have a turn.

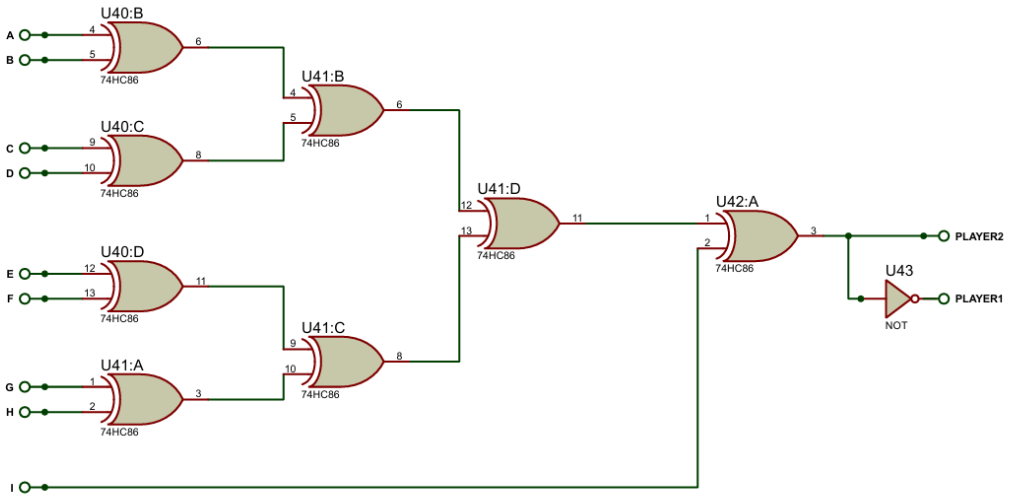
Profile Generator:

The Profile Generator circuit determines whose turn it is in the game. It consists of XOR gates, and its operation ensures that players alternate turns seamlessly. When no player has made a move, all inputs remain at 0. In this case, the PLAYER-2-PROFILE output is 0, and the PLAYER-1-PROFILE output is 1, as the outputs are inverted.

When Player 1 plays (an input goes high), the XOR gate processes the change, setting PLAYER-2-PROFILE to 1 and PLAYER-1-PROFILE to 0. This alternation continues with every input change, toggling the profiles between the players to enforce turn-based gameplay.

For simplicity, the design uses 4 inputs instead of 9, and it has two outputs: PLAYER-1-PROFILE and PLAYER-2-PROFILE. The circuit's behavior guarantees proper toggling with each player's action, ensuring fairness and consistency in the game.

- ## Schematic:



- **Simplified Truth Table:**

A	B	C	D	PLAYER-2	PLAYER-1
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	1

- **Equations:**

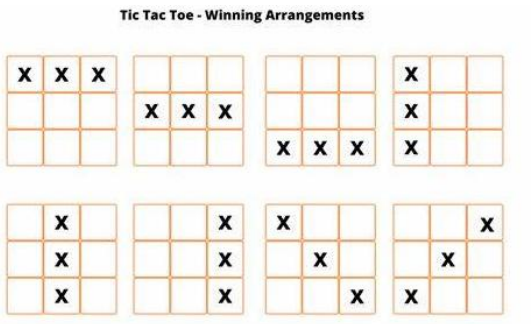
- **PLAYER2 = (A ⊕ B ⊕ C ⊕ D)**
- **PLAYER1 =$\overline{(A \oplus B \oplus C \oplus D)}$**

Extending the result to all the 9 inputs we have:

- **PLAYER 2 = (A ⊕ B ⊕ C ⊕ D ⊕ E ⊕ F ⊕ G ⊕ H ⊕ I)**
- **PLAYER1 =$\overline{(A \oplus B \oplus C \oplus D \oplus E \oplus F \oplus G \oplus H \oplus I)}$**

• Winning Conditions:

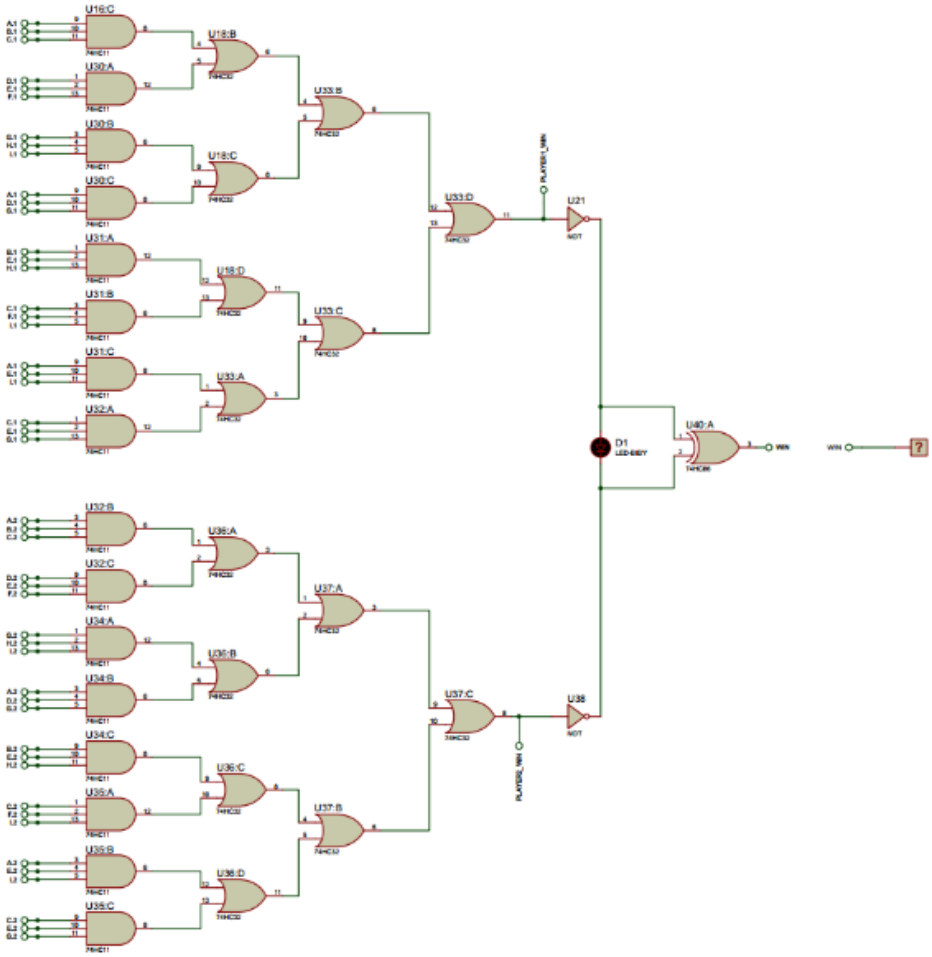
In **Tic Tac Toe**, a player wins when they successfully align their symbols (e.g., "X" or "O") in one of the following ways:



- A **Row** (3 aligned horizontally).
- A **Column** (3 aligned vertically).
- A **Diagonal** (3 aligned diagonally).

The game ends immediately once a player meets one of these conditions.

• Full schematic:

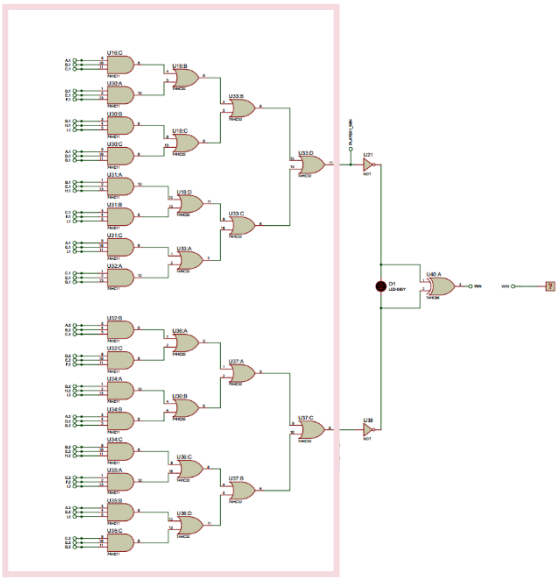


To determine the winning condition in hardware, the values of each possible triplet are connected to **3-input AND gates**. These gates output **high (1)** when all three connected inputs are high (indicating alignment).

Since a winner is found when at least one of these gates outputs high, the outputs of the AND gates are connected to an **OR gate**. This ensures that the circuit outputs **high** if any one of the eight possible winning combinations is satisfied.

The function that determines if a winner is found is:

W=ABC+*D*EF+*GHI*+A*DG*+BEH+CFI+AEI+CEG



To identify whether the game is won by Player 1 or Player 2, the circuit is duplicated for each player.

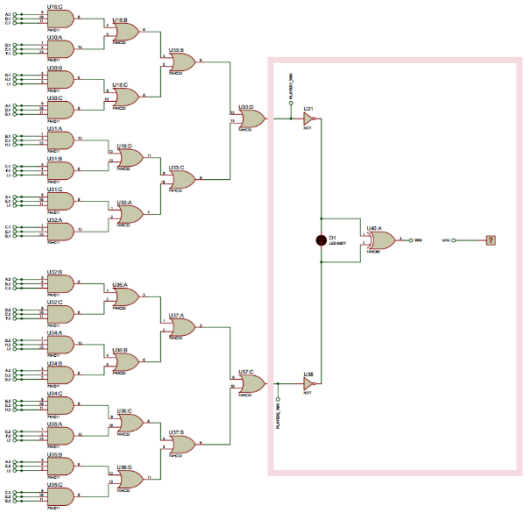
- For **Player 1**, the inputs correspond to their moves (A1, B1, C1, ...A1, B1, C1, ...A1, B1, C1, ...).
- For **Player 2**, the inputs correspond to their moves (A2, B2, C2, …A2, B2, C2, ...A2, B2, C2....).

The winning functions for each player are:

W1 = A1B1C1+*D*1E1F1+*G*1*H*1*I*1+A1*D*1*G*1+B1E1H1+C1F1I1+A1E1I1+C1E1G1

W2 = A2B2C2+*D*2E2F2+*G*2*H*2*I*2+A2*D*2*G*2+B2E2H2+C2F2I2+A2E2I2+C2E2G2

A new variable, WIN, is defined to indicate if the game has a winner, regardless of whether it's Player 1 or Player 2.



This is achieved using an XOR gate and NOT gate to ensure mutual exclusivity:

- WIN is **1** if either W1 or W2 is high (indicating a winner).
- WIN is **0** if neither W1 nor W2 is high, or if both are high (invalid state).

It is defined as follows:

WIN = $\overline{\mathbf{W1}} \oplus \overline{\mathbf{W2}}$

This new variable is equal to 1 only when one of the two players has won the game.

Check Draw Circuit:

There is a chance that all switches are pressed and there is no winner; then it is a draw. For this another circuit is connected to the switches, where it takes the output of all the switches and uses AND logic gates. When all switches are pressed the value of ALL_PRESSED will be high.

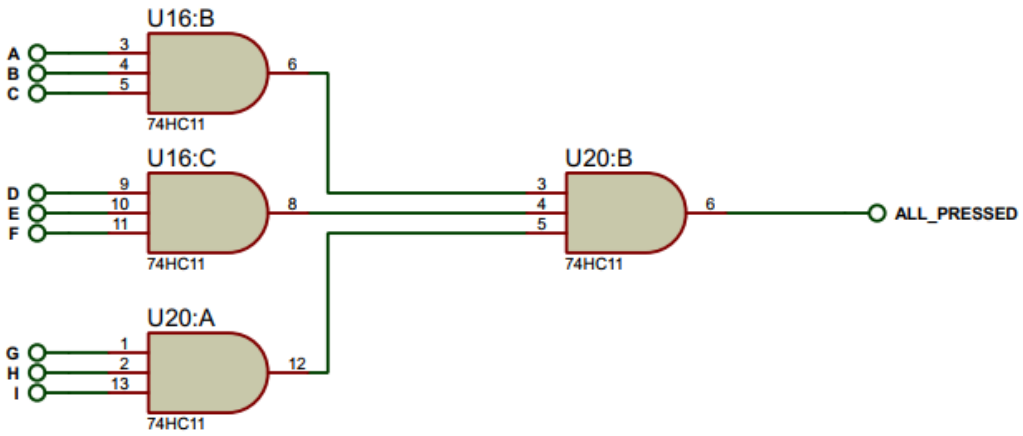
We check the no win condition by seeing if the output of both players win is low.

By reversing the win output to high and checking that ALL_PRESSED is also high our DRAW output will be high.

Here we used 3-input AND gates, 2-input AND gate and an OR gate.

- **Truth Table:**

A	B	C	D	E	F	G	H	I	ALL_PRESSED
0	X	X	X	X	X	X	X	X	0
X	0	X	X	X	X	X	X	X	0
X	X	0	X	X	X	X	X	X	0
X	X	X	0	X	X	X	X	X	0
X	X	X	X	0	X	X	X	X	0
X	X	X	X	X	0	X	X	X	0
X	X	X	X	X	X	0	X	X	0
X	X	X	X	X	X	X	0	X	0
X	X	X	X	X	X	X	X	0	0
1	1	1	1	1	1	1	1	1	1

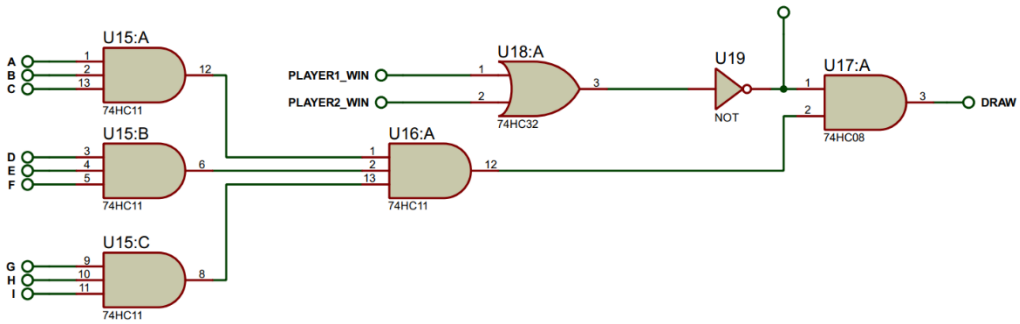


- $\mathbf{ALL\_PRESSED = A \cdot B \cdot C \cdot D \cdot E \cdot F \cdot G \cdot H \cdot I}$

PLAYER1_WIN	PLAYER2_WIN	ALL_PRESSED	DRAW
0	0	0	0
0	0	1	1
0	1	X	0
1	0	X	0

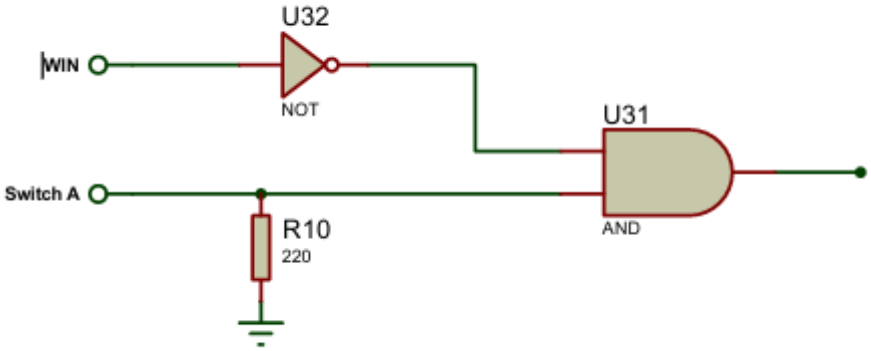
- $\mathbf{DRAW = \overline{PLAYER1\_WIN + PLAYER2\_WIN} \cdot ALL\_PRESSED}$

FULL DRAW CIRCUIT:



Blocking the game once a winner is found:

To block the game once a winner is determined, we used an AND gate connected to the input of each switch and the winning signal, followed by a NOT gate. This configuration ensures that as long as a winner has not been found, both inputs to the AND gate will remain high (when a button is pressed), allowing the output to pass through. However, once a winner is identified, the AND gate output is blocked, preventing players from lighting the LEDs further.



Game Play:

This setup represents what the user actually interacts with. Buttons are used to make a move, and LEDs light up in different colors to indicate which player's turn it is. A reset button allows the game to be restarted, while two additional LEDs display the current winning status.